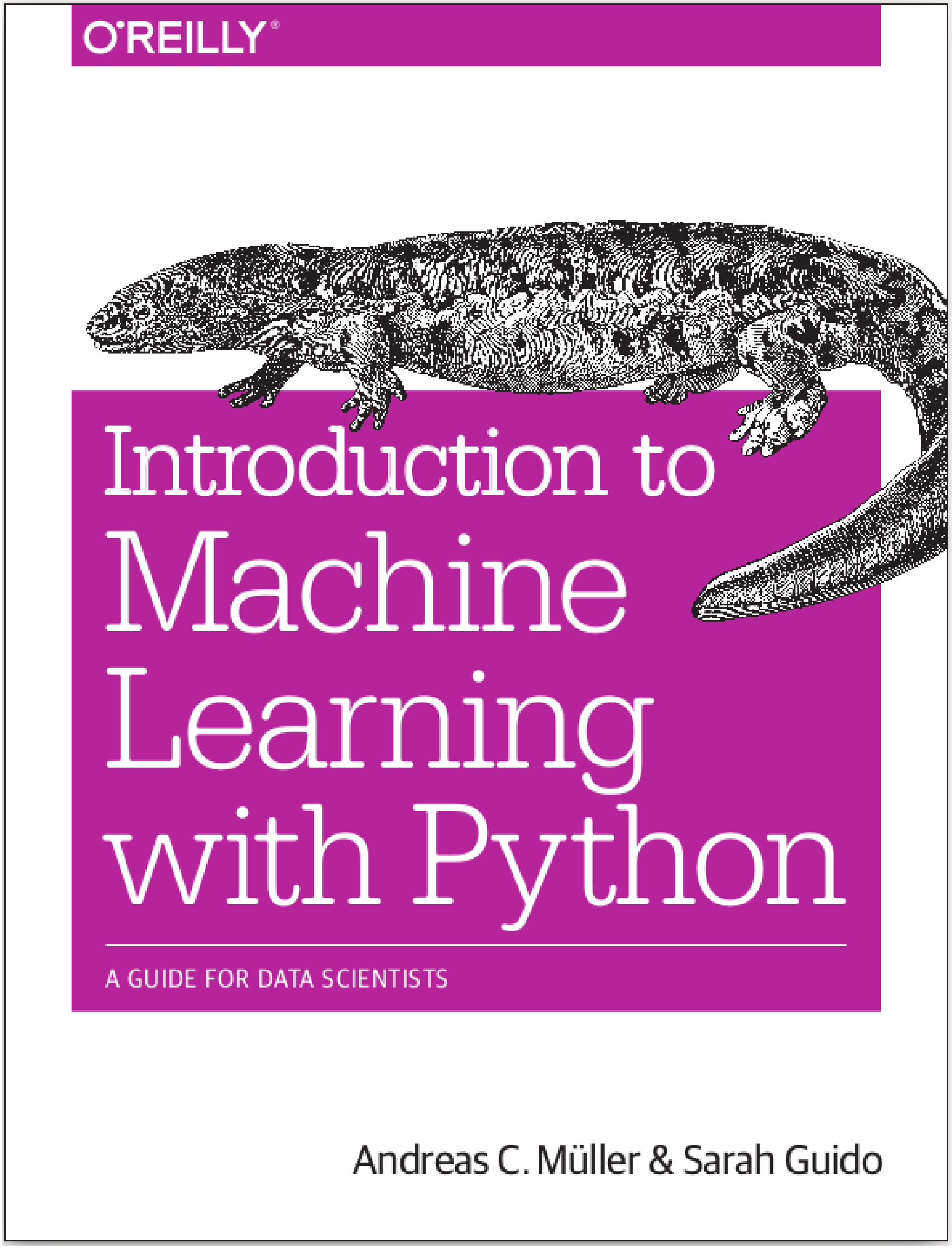
Writing a book with Jupyter Notebooks

Andreas Müller
Columbia University, scikit-learn

Alfred P. Sloan
FOUNDATION

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK



O'REILLY®

Introduction to Machine Learning with Python

A GUIDE FOR DATA SCIENTISTS

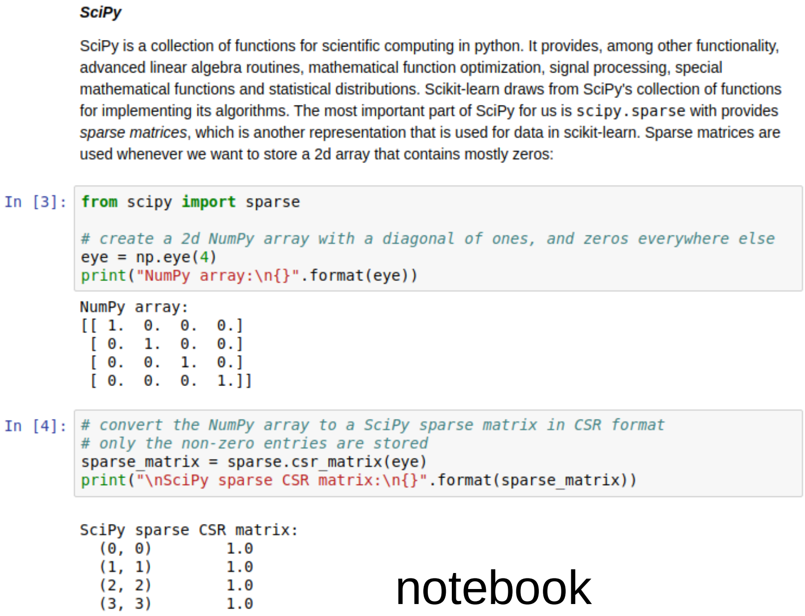
Andreas C. Müller & Sarah Guido

Book signing and giveaway today 3:20!

Not about bookbook

https://github.com/takluyver/bookbook

By Thomas Kluyver



SciPy

SciPy is a collection of functions for scientific computing in python. It provides, among other functionality, advanced linear algebra routines, mathematical function optimization, signal processing, special mathematical functions and statistical distributions. Scikit-learn draws from SciPy's collection of functions for implementing its algorithms. The most important part of SciPy for us is `scipy.sparse` with provides *sparse matrices*, which is another representation that is used for data in scikit-learn. Sparse matrices are used whenever we want to store a 2d array that contains mostly zeros:

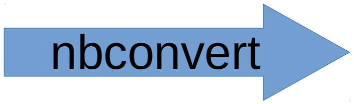
```
In [3]: from scipy import sparse

        # create a 2d NumPy array with a diagonal of ones, and zeros everywhere else
        eye = np.eye(4)
        print("NumPy array:\n{}".format(eye))
```

```
NumPy array:
[[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]
```

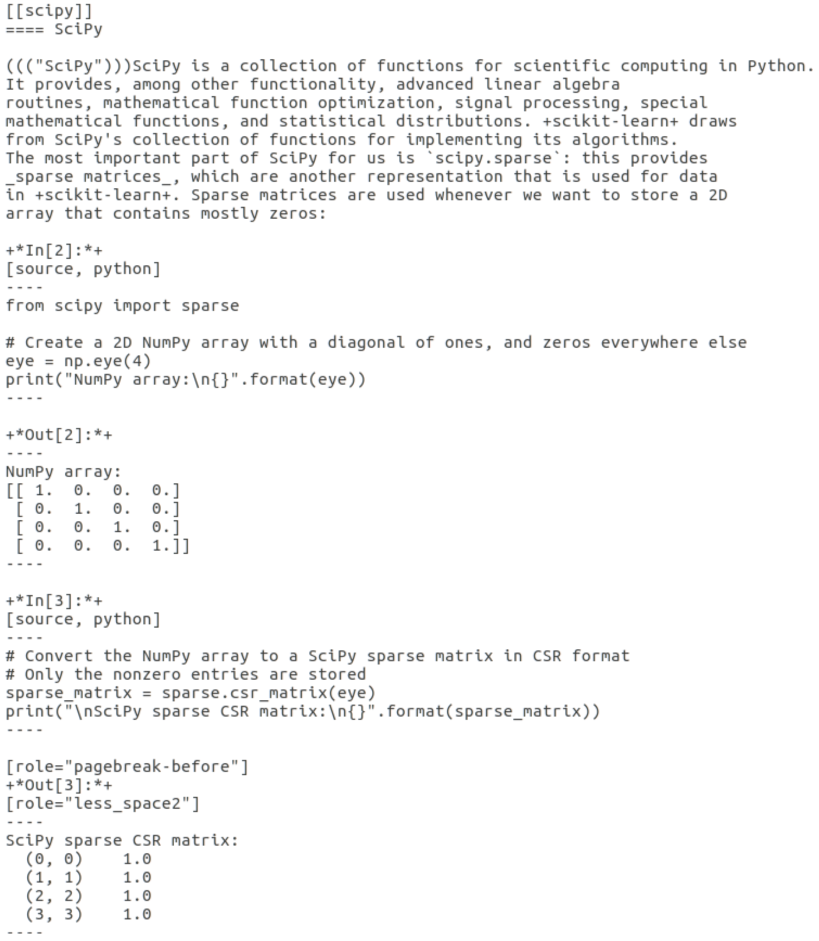
```
In [4]: # convert the NumPy array to a SciPy sparse matrix in CSR format
        # only the non-zero entries are stored
        sparse_matrix = sparse.csr_matrix(eye)
        print("\nSciPy sparse CSR matrix:\n{}".format(sparse_matrix))
```

```
SciPy sparse CSR matrix:
  (0, 0)	1.0
  (1, 1)	1.0
  (2, 2)	1.0
  (3, 3)	1.0
```

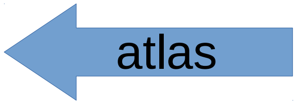
notebook

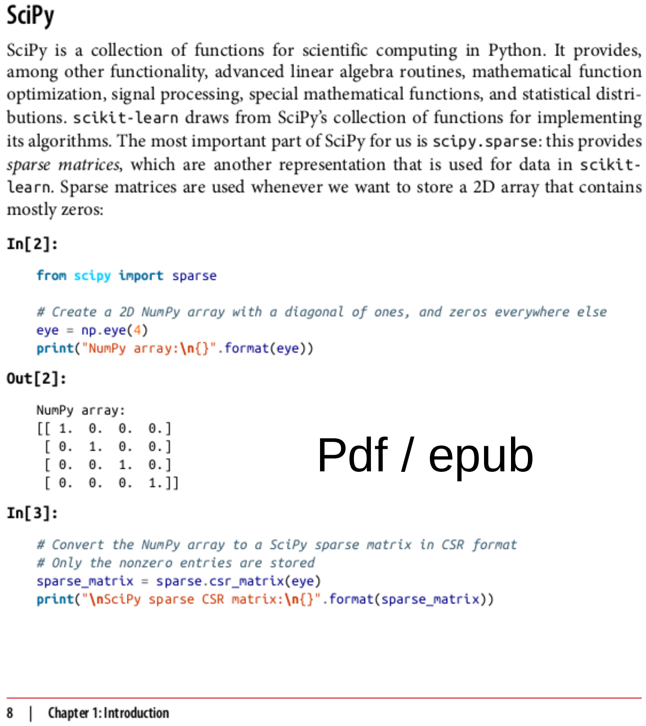


asciidoc



```
[[scipy]]
==== SciPy

((("SciPy")))SciPy is a collection of functions for scientific computing in Python.
It provides, among other functionality, advanced linear algebra
routines, mathematical function optimization, signal processing, special
mathematical functions, and statistical distributions. +scikit-learn+ draws
from SciPy's collection of functions for implementing its algorithms.
The most important part of SciPy for us is `scipy.sparse`: this provides
_sparse matrices_, which are another representation that is used for data
in +scikit-learn+. Sparse matrices are used whenever we want to store a 2D
array that contains mostly zeros:

+*In[2]:*+
[source, python]
----
from scipy import sparse

# Create a 2D NumPy array with a diagonal of ones, and zeros everywhere else
eye = np.eye(4)
print("NumPy array:\n{}".format(eye))
----

+*Out[2]:*+
----
NumPy array:
[[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]
----

+*In[3]:*+
[source, python]
----
# Convert the NumPy array to a SciPy sparse matrix in CSR format
# Only the nonzero entries are stored
sparse_matrix = sparse.csr_matrix(eye)
print("\nSciPy sparse CSR matrix:\n{}".format(sparse_matrix))
----

[role="pagebreak-before"]
+*Out[3]:*+
[role="less_space2"]
----
SciPy sparse CSR matrix:
  (0, 0)    1.0
  (1, 1)    1.0
  (2, 2)    1.0
  (3, 3)    1.0
----
```





SciPy

SciPy is a collection of functions for scientific computing in Python. It provides, among other functionality, advanced linear algebra routines, mathematical function optimization, signal processing, special mathematical functions, and statistical distributions. `scikit-learn` draws from SciPy's collection of functions for implementing its algorithms. The most important part of SciPy for us is `scipy.sparse`: this provides *sparse matrices*, which are another representation that is used for data in `scikit-learn`. Sparse matrices are used whenever we want to store a 2D array that contains mostly zeros:

In[2]:

```
from scipy import sparse

# Create a 2D NumPy array with a diagonal of ones, and zeros everywhere else
eye = np.eye(4)
print("NumPy array:\n{}".format(eye))
```

Out[2]:

```
NumPy array:
[[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]
```

Pdf / epub

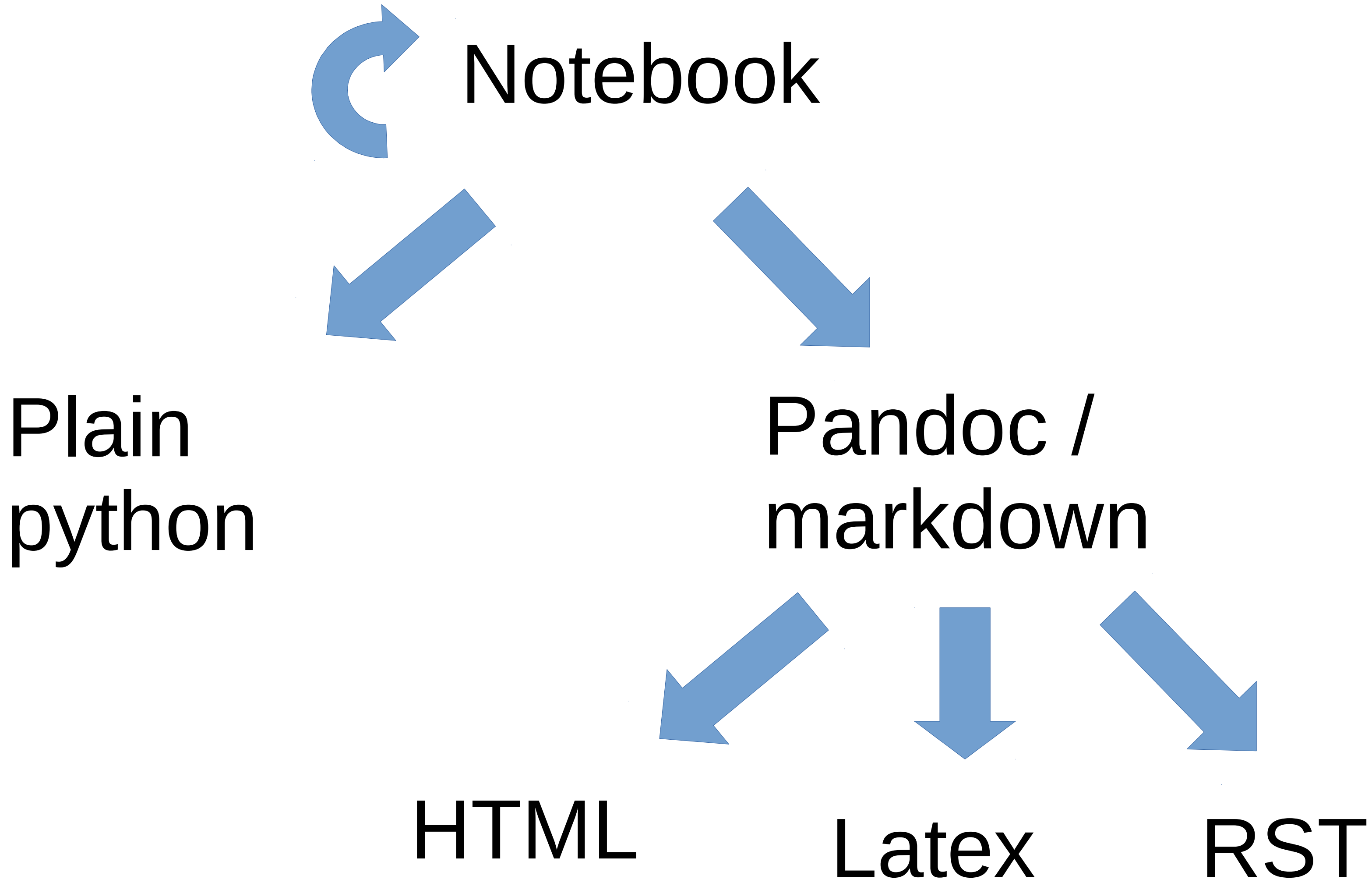
In[3]:

```
# Convert the NumPy array to a SciPy sparse matrix in CSR format
# Only the nonzero entries are stored
sparse_matrix = sparse.csr_matrix(eye)
print("\nSciPy sparse CSR matrix:\n{}".format(sparse_matrix))
```

8 | Chapter 1: Introduction

nbconvert

nbconvert



Notebook

Plain python

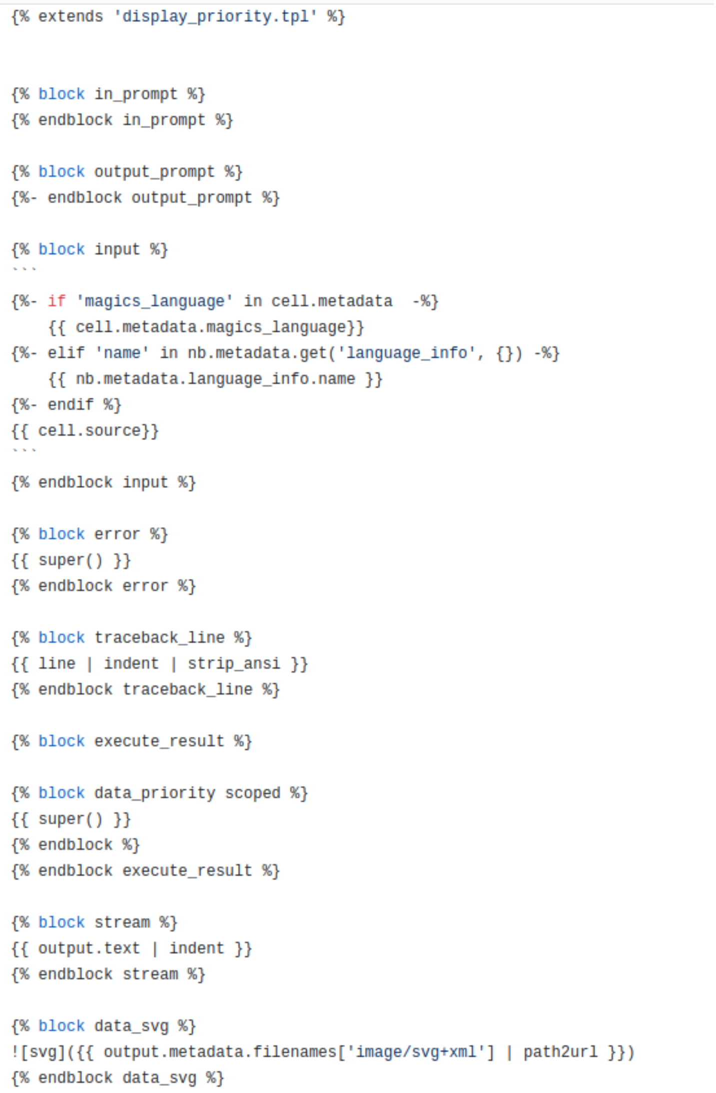
Pandoc / markdown

HTML

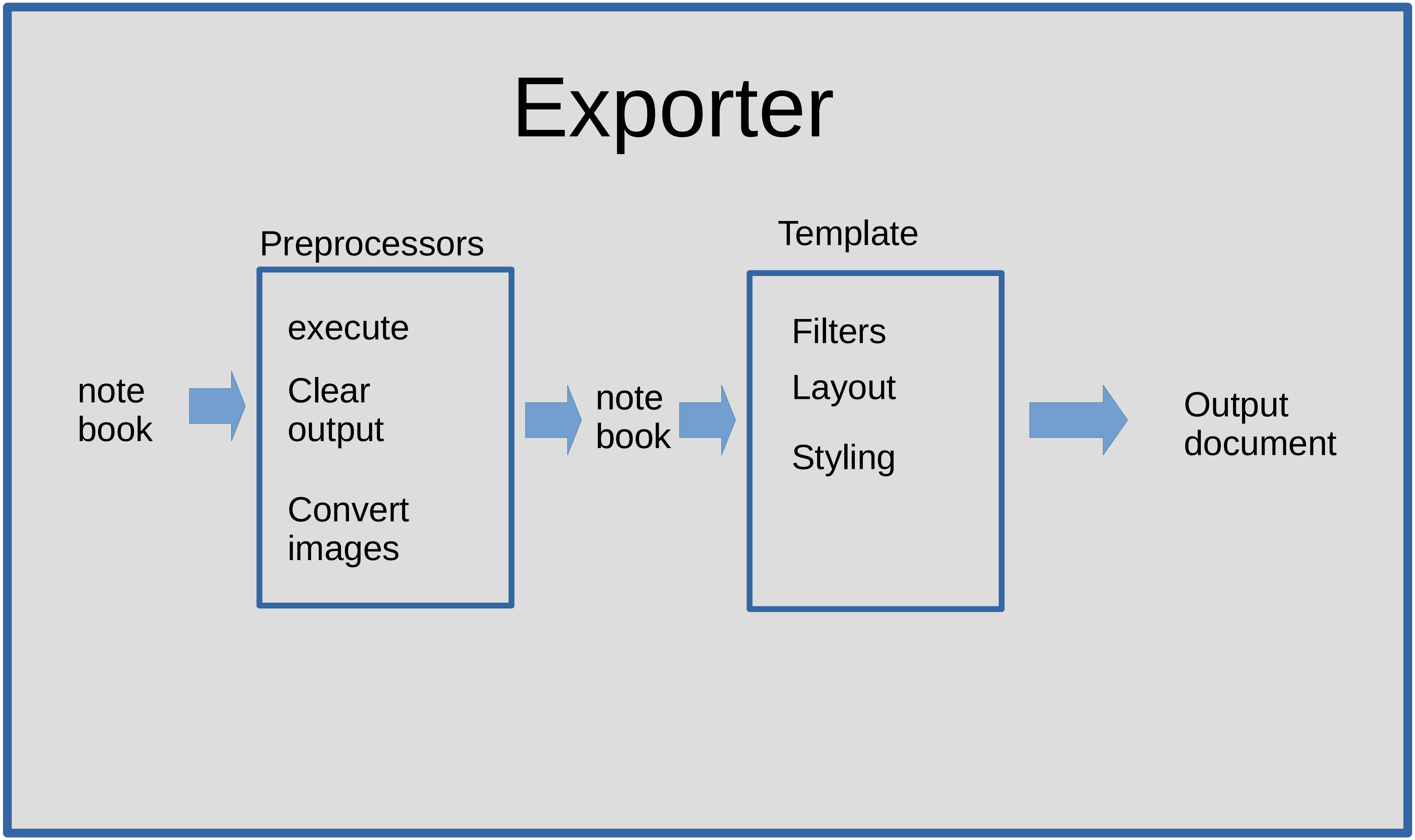
Latex

RST

via Jinja



```
{% extends 'display_priority.tpl' %}


{% block in_prompt %}
{% endblock in_prompt %}

{% block output_prompt %}
{%- endblock output_prompt %}

{% block input %}
```
{%- if 'magics_language' in cell.metadata -%}
    {{ cell.metadata.magics_language}}
{%- elif 'name' in nb.metadata.get('language_info', {}) -%}
    {{ nb.metadata.language_info.name }}
{%- endif %}
{{ cell.source}}
```
{% endblock input %}

{% block error %}
{{ super() }}
{% endblock error %}

{% block traceback_line %}
{{ line | indent | strip_ansi }}
{% endblock traceback_line %}

{% block execute_result %}

{% block data_priority scoped %}
{{ super() }}
{% endblock %}
{% endblock execute_result %}

{% block stream %}
{{ output.text | indent }}
{% endblock stream %}

{% block data_svg %}
![svg]({{ output.metadata.filenames['image/svg+xml'] | path2url }})
{% endblock data_svg %}
```



Exporter

note book

Preprocessors

execute

Clear output

Convert images

note book

Template

Filters

Layout

Styling

Output document

(left out postprocessors and writers that are less important for customization)

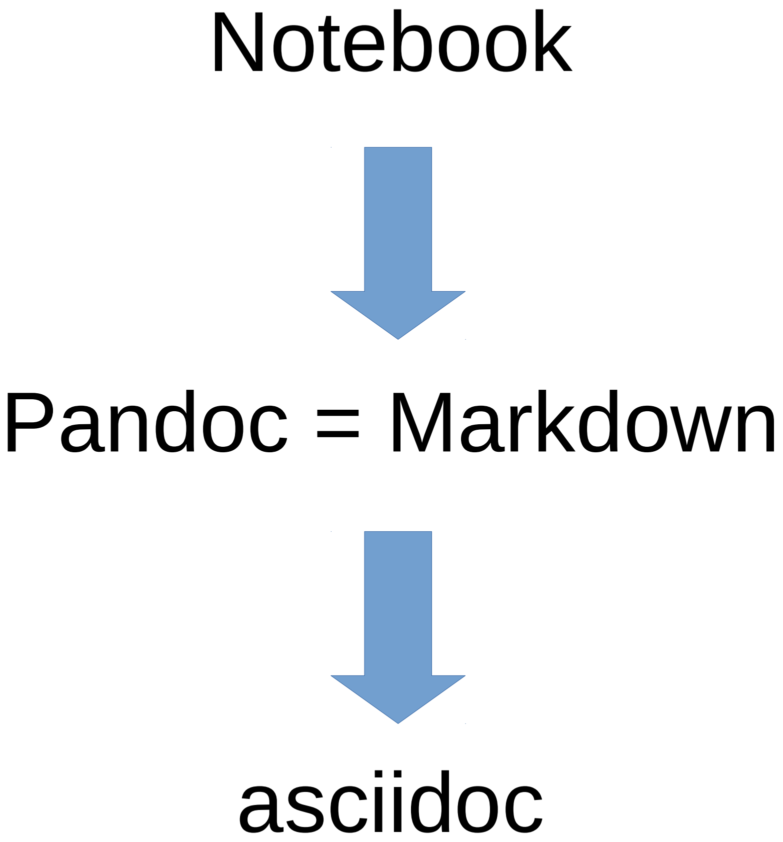
http://nbconvert.readthedocs.io/en/stable/architecture.html Thanks Min!

Adding an output format: asciidoc

- Add asciidoc exporter
- Add asciidoc template
- Add filters for converting to asciidoc (via pandoc)

- Add asciidoc exporter
- Add asciidoc template
- Add filters for converting to asciidoc (via pandoc)

easy!



Notebook

Pandoc = Markdown

asciidoc

Limitations of Pandoc / Markdown

No Colspan / Rowspan in Tables

No internal references
(to sections, figures, equations)

No references across documents.

No figure captions.
(Also hard to express in notebook)

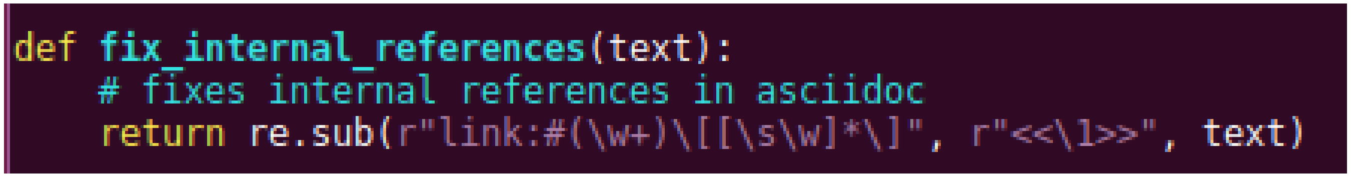
Html to rescue?

Pandoc passes through html
when converting to html.

Pandoc strips html in any other case!

Overcoming Limitations

Add internal references via filters:



```
def fix_internal_references(text):
    # fixes internal references in asciidoc
    return re.sub(r"link:#(\w+)\[[\s\w]*\]", r"<<\1>>", text)
```

![anythinghere](#internal_reference) becomes internal reference.

Reference points currently done manually (by editors).

Figure captions / colspan:

?!?!

(ideas welcome for my next book ;)

Lessons

Writing exporters is easy!
Writing great is exporters is hard.

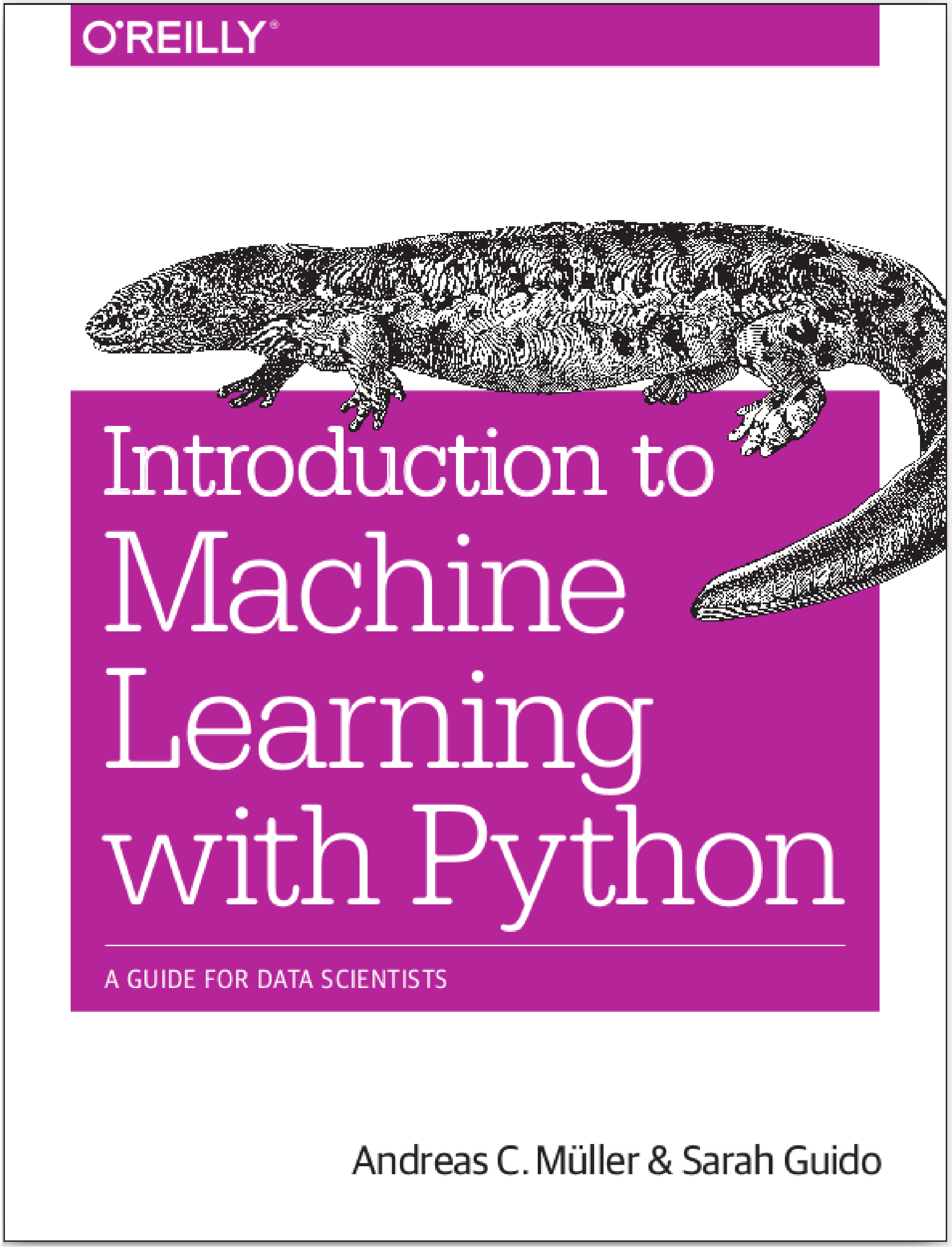
Pandoc is not expressive enough!

Preprocessors + Jinja templates + filters
allow you to do anything you want!

Keep notebooks short – split up chapters!

Unsolved problem

Upstreaming asciidoc edits into Notebooks.
Probably possible.



O'REILLY

Introduction to Machine Learning with Python

A GUIDE FOR DATA SCIENTISTS

Andreas C. Müller & Sarah Guido

amueller.github.io

@amuellerml

@amueller

t3kcit@gmail.com

https://github.com/amueller/talks_odt/